情報保護學會論文誌
第13券 第6號, 2003. 12

공개키 기반의 공모자 추적기법에서의 추적 임계치에 관한 연구

임 정 미*†, 이 병 선*, 박 창 섭*‡

A Study on Tracing-Threshold of Public-Key Traitor-Tracing Schemes

Jeung-Mi Rim*†, Byoung-Sun Lee*, Chang-Seop Park*‡

요 약

공모자 추적기법에서의 임계치는 추적기법에 의해서 신원이 밝혀질 수 있는 공모자들의 최대 수를 의미한다. 본 논문의 대상이 되는 선형오류수정부호를 이용한 공모자 추적기법에서는 기반이 되는 오류수정부호의 오류수정능력에 의해서 임계치가 결정된다. 본 연구에서는 공모의 규모가 임계치를 넘었을 경우에 발생하는 현상을 추적기법의 조합론적인 특성을 중심으로 분석하고, 이를 기반으로 동일한 불법 복호화 키를 만들어 낼 수 있는 두 개의 상호 독립적인 사용자 그룹의 존재 가능성을 보인다.

ABSTRACT

The threshold value of the traitor-tracing schemes means a maximum number of traitors whose identities can be uniquely exposed using the tracing scheme. In the traitor-tracing scheme based on an error-correcting code, which is focused at this paper, the threshold value is determined by the error-correcting capability of the underlying error-correcting code. Analyzed in terms of a combinatorial property of the tracing scheme is the resulting effect on the tracing scheme when the collusion size is over the threshold value, and a possibility of two disjoint groups of users making an identical unauthorized decryption key is shown.

keyword :

Ⅰ. 서 론

유료 TV 방송, 온라인 데이터베이스, 그리고 소프트웨어 분배와 같은 응용분야에서 인가되지 않은 사용자에 대한 접근 통제를 하기 위해서는, 데이터 공급자(data supplier)는 데이터에 접근할 수 있는 암호화 된 세션키(encrypted session key)를 브로드캐스트 또는 멀티캐스트 방식으로 전송하고, 인가된 사용자(authorized user)들은 자신의 개별 복호화 키(personal decryption key)를 이용하여 세션키를 획득할 수 있다. 하지만, 문제는 몇명의 인가된 사용자들이 공모하여 그들의 개별 복호화 키를 인가되지 않은 사용자들(pirates)에게 전달 해 줌으로써, 그들이 불법적인 복호기(decoder)를 제작하여 유포할 수 있는 가능성을 제공할 수 있다는 데에 있다. 이러한 문제점을 해결할 수 있는 방법 중의 하나는 공모에 참여한 즉, 자신의 개별 복호화 키를 제공한 사용자의 신원 파악을 압수된 불법적인 복호기에 대한 분석을 통해 가

* 이 연구는 2003학년도 단국대학교 대학연구비의 지원으로 연구되었습니다.
** 단국대학교 전자계산학과({redpig3, csp0}@dankook.ac.kr, nokdu76@hanmail.net)
† 주저자, ‡ 교신저자, 논문접수일 : 2003년 8월 11일, 심사완료일 : 2003년 12월 1일

능하게 함으로써, 인가된 사용자들이 자신의 키를 다른 사용자에게 유포하는 것을 사전에 억제하는 것이다. 이러한 공모자 추적기법(traitor tracing scheme)은 Chor와 Naor[3]에 의해서 처음 제시 되었는데, 기본적인 개념은 개별 복호화 키의 원래 소유자를 추적할 수 있게 각각의 인가된 사용자들의 키에 식별(fingerprinting) 기능을 내장 시키는 것이다.

최근, Reed-Solomon 부호를 이용한 공개키 기반의 공모자 추적기법이 Boneh와 Franklin[2]에 의해서 제안되었다. 그들이 제안한 공개키 기반의 브로드캐스트 암호화 기법의 특징은, 암호화를 위한 한 개의 공개키가 주어졌을 때, 복호화에 소요될 각 사용자들의 서로 다른 개인키는 t개의 오류를 수정할 수 있는 Reed-Solomon 부호를 통해서 생성된다는 데에 있다. 이렇게 생성된 개인키는 개별 복호화 키를 의미하는데, 이를 기반으로 최대 t명의 공모자들에 의해서 제작되어 불법 복호기 안에 내장된 각각의 공모자들의 개인키가 식별되어 공모자들을 색출할 수 있게 된다. 여기서 임계치 t는 Reed-Solomon 부호의 오류수정능력(error-correcting capability)으로부터 도출되어진다. 또한, 그들은 t+1명 이상 (그러나 2t 명 미만)의 공모자가 공모했을 경우에도 Guruswami와 Sudan[4]에 의해서 제안된 리스트 복호 알고리즘(List Decoding Algorithm)을 이용하여 일부 공모자들에 대한 부분적인 신원 파악이 가능함도 언급하였다.

본 논문에서는 t+1명 이상의 공모자가 공모했을 경우에 대한 내용을 조합론적인 측면에서 분석한다. t+1명 이상의 공모 발생은 결국 오류수정부호의 복호화 과정에서 복호화 오류(decoding error)가 발생한다는 것이고, 따라서 다른 일부 무고한 사용자가 공모자로 오인될 가능성이 있다. 이와 관련하여, 특정 공모자 그룹 U에 의해서 만들어진 불법적인 복호화 키가 주어졌을 때, 이와 동일한 키 값을 만들어낼 수 있는 다른 사용자 그룹들 V의 존재 여부, 그리고 존재한다면 어떠한 방식으로 동일한 키 값을 만들어 낼 수 있는지 등을 분석한다. 특히, $U \cap V = \varnothing$ 조건을 만족하는 그룹에 초점을 맞춤으로써 t+1명 이상의 공모가 발생했을 경우에 이들 공모자 그룹 U와는 독립적인 별개의 t명 이하의 무고한 사용자 그룹이 공모자로 오인될 수 있음을 보인다. 2장에서는 Boneh와 Franklin의 기법을 수정한 공개키 기반의 공모자 추적 시스템[5]에 대해서 소개를 하고, 3장에서는 1개 이상의 그룹에 속할 수 있는 불법적인 개별 복호화 키의 구축방식을 소개함으로써 무고한 사용자들이 어떻게 공모자로 오인될 수 있는지를 보인다.

II. 선형부호를 이용한 공개키 기반 공모자 추적 기법

이번 장에서는 본 논문에서 사용되는 기호와 용어를 정의하는 의미에서 오류수정부호에 대한 간략한 내용과, 이에 기반을 두고 제안되어진 공모자 추적기법에 대해 소개한다. 특히, 여기서 소개되는 공모자 추적기법은 원래의 Boneh-Franklin 방식을 간소화한 기법으로써, 3장의 분석에서도 이 간소화된 Boneh-Franklin 방식을 이용한다.

2.1 오류수정부호

소수 q에 대해서 C를 t개의 오류를 수정할 수 있는 GF(q) 상의 (n, k) 선형부호(linear code), 그리고 $\mathbf{G}$와 $\mathbf{H}$를 각각 이에 대응되는 k x n 생성행렬(generator matrix), (n k) x n 패러티 검사행렬(parity-check matrix)이라고 하자. GF(q) 상의 메시지 벡터 $\boldsymbol{m}$은 생성행렬을 통해서 코드워드(codeword) $\boldsymbol{c} = \boldsymbol{m} \cdot \mathbf{G}$로 부호화된다. 특히, 생성행렬과 패러티 검사행렬 간에는 $\mathbf{G} \cdot \mathbf{H}^T = \mathbf{0}$이 성립하기 때문에 $\boldsymbol{c} \cdot \mathbf{H}^T = \mathbf{0}$가 된다. 채널 상에서 발생되는 오류벡터(error vector) $\boldsymbol{e}$는 코드워드에 첨가되고 수신자는 수신된 $\boldsymbol{c}+\boldsymbol{e}$를 패러티 검사행렬을 기반으로 복호화 작업을 수행한다. 만약, 오류벡터 $\boldsymbol{e}$의 해밍무게(Hamming weight)가 t이하일 경우에는 계산된 오증벡터(syndrome vector) $\boldsymbol{s} = (\boldsymbol{c}+\boldsymbol{e}) \cdot \mathbf{H}^T = \boldsymbol{e} \cdot \mathbf{H}^T$으로부터 오류벡터를 도출하게 되어 오류를 수정할 수 있게 된다.

오증벡터로부터 오류벡터를 도출해 내기 위한 간단한, 하지만 비효율적인 방식은 표준배열(standard array)을 이용하는 것이다. 즉, 표준배열은 수정 가능한 오류벡터와 그에 대응되는 오증벡터들로 구성된 표를 의미한다. 표준배열은 GF(q)상의 q^n개의 벡터를 q^{n-k} 개의 coset으로 분할을 한다. 또한, 각각의 coset은 q^k개의 벡터로 구성된다. $\boldsymbol{c}_1$을 영 코드워 드(zero codeword)라 정의하면, 첫번째 coset $\{ \boldsymbol{c}_1, \boldsymbol{c}_2, \cdots, \boldsymbol{c}_{qk} \} = \{ \mathbf{0}, \boldsymbol{c}_2, \cdots, \boldsymbol{c}_{qk} \}$은 선형부호 C의 모든 가능한 코드워드를 포함하게 되고, 그 이후의 x번째 coset ($2 \le x \le q^{n-k}$)은 $\{ \boldsymbol{e}_x, \boldsymbol{c}_2 + \boldsymbol{e}_x, \cdots, \boldsymbol{c}_{qk} + \boldsymbol{e}_x \}$와 같이 구성이 된다. 여기서, $\boldsymbol{e}_x$ 는 coset leader, 즉 수정 가능한 오류벡터를 의미하는데, 이전의 x - 1개의 coset들을 구축하는 데에 사용이 안된 나머지 벡터들 중에

서 해밍무게가 가장 작은 벡터로 선정이 된다[1].

특정 오류수정부호가 가지는 오류수정 능력 t는 그 부호의 최소거리 $d \geq 2t+1$에 의해서 결정된다. 이 최소거리의 의미는 t개의 오류를 수정할 수 있는 특정 부호에 존재하는 모든 코드워드 중에서 임의의 2개의 코드워드를 선정하여 해밍무게를 검사했을 경우에 그 값이 적어도 $2t+1$이상이 됨을 의미한다.

2.2 공개키 기반 공모자 추적 기법

Kurosawa와 Yoshida[5]는 Boneh-Franklin 기법에서의 개별 복호화 키 생성방식이 필요 이상의 복잡한 구성방식을 채택하여 공개키 암호화 및 복호화에 불필요한 부하를 가중시킨다고 주장하고, 이를 간소화한 기법을 제안하였다. 하지만, Boneh-Franklin 기법에서의 추적 알고리즘은 그대로 유지하였다. 본 절에서는 Kurosawa와 Yoshida가 제안한 기법을 중심으로 설명을 한다.

데이터 공급자는 시스템 전반적으로 사용할, t개의 오류를 수정할 수 있는 GF(q) 상의 (n, k) 선형부호의 패러티 검사행렬 $\mathbf{H}$와, 시스템 비밀키 $\boldsymbol{a} = (a_1, a_2, ..., a_{n-k})$, $a_j \in_R Z_q$를 선정하고, 이를 기반으로 사용자 $i \in \{1, 2, ..., n\}$의 개별 복호화 키 $b_i = (a_1, a_2, ..., a_{n-k}) \otimes \boldsymbol{h}_i$를 계산하여 안전하게 사용자 i에게 전달한다. 이때, $\boldsymbol{h}_i = (h_{1i}, h_{2i}, ..., h_{(n-k)i})$는 행렬 $\mathbf{H}$의 i번째 열(column)이고 '$\otimes$'는 2개 벡터간 내적(inner product)을 의미한다. 또한, g를 Z_q의 생성자라 할 때, 공개키($y_1=g^{a1}$, $y_2=g^{a2}$, ..., $y_{n-k}=g^{an-k}$)를 $\mathbf{H}$와 함께 공개한다. 세션키 k_s를 암호화하기 위해서 데이터 공급자는 먼저, 임의의 난수 $r \in_R Z_q$을 생성한 후에($k_s \cdot g^r$, y_1^r, y_2^r, ..., y_{n-k}^r)을 계산하여 브로드캐스트 한다. 이를 전송 받은 사용자 i는 자신이 계산한 $b_i^{-1} \cdot \boldsymbol{h}_i$을 기반으로 다음과 같이 복호화 과정을 수행하여 세션키 k_s를 획득한다.

$$k_s = k_s \cdot g^r / \prod_{j=1}^{n-k} (y_j^r)^{b_i^{-1} \cdot h_{ji}}$$
$$= k_s \cdot g^r / (g^r)^{(\sum_{j=1}^{n-k} a_j h_{ji}) \cdot b_i^{-1}} \quad (1)$$

$\tau (\leq t)$명의 공모자들 $i_1, i_2, ..., i_\tau$ 이 공모하여 불법 복호화 키 $s = (s_1, s_2, ..., s_{n-k}) = \beta_1(b_{i1}^{-1}) \cdot \boldsymbol{h}_{i1} + \beta_2(b_{i2}^{-1}) \cdot \boldsymbol{h}_{i2} + ... + \beta_\tau(b_{i\tau}^{-1}) \cdot \boldsymbol{h}_{i\tau}$를 내장한 불법 디코더를 만들었다고 가정하자. 이때, $\beta_1 + \beta_2 + ... + \beta_\tau = 1$. 이 불법적인 복호화 키를 이용한 세션키 획득과정은 다음과 같다.

$$k_s = k_s \cdot g^r / \prod_{j=1}^{n-k} (y_j^r)^{s_j}$$
$$= k_s \cdot g^r / (g^r)^z \text{ , where}$$
$$z = \beta_1 (\sum_{j=1}^{n-k} a_j h_{ji_1}) \cdot b_{i_1}^{-1} + \beta_2 (\sum_{j=1}^{n-k} a_j h_{ji_2}) \cdot b_{i_2}^{-1} + ... + \beta_\tau (\sum_{j=1}^{n-k} a_j h_{ji_\tau}) \cdot b_{i_\tau}^{-1} \quad (2)$$

불법적인 복호기 안에 내장된 s가 압수되었을 때, 그 공모에 참여한 각각의 공모자를 색출해내기 위한 공모자 추적 알고리즘은, 결국 s를 생성하는 데에 이용된 공모자들의 개별 복호화 키 $\{b_{i1}, b_{i2}, ..., b_{i\tau}\}$를 식별해 내는 것이다. 실제로, 개별 복호화 키 b_{il} ($l = 1, 2, \cdots, \tau$)은 시스템 비밀키에 행렬 $\mathbf{H}$의 해당 열 벡터 $\boldsymbol{h}_{il}$을 곱해서 만들어지기 때문에 그 열 벡터의 위치 값 i_l을 파악하는 것으로 충분하다.

여기서의 공모자 추적 알고리즘은 오류수정부호의 복호화 과정(decoding process)에 기반을 두고 있다. 오류수정부호의 복호화 과정과 관련하여, s는 오류벡터(error vector) $\boldsymbol{e} = (e_1, e_2, \cdots, e_n)$에 대응되는 오증벡터(syndrome vector)로 간주되어질 수 있다.

$$s = \boldsymbol{e} \cdot \mathbf{H}^T = e_1 \cdot \boldsymbol{h}_1 + e_2 \cdot \boldsymbol{h}_2 + ... + e_n \cdot \boldsymbol{h}_n \quad (3)$$

만약 C가 BCH 또는 Reed-Solomon 부호와 같은 순환부호(cyclic code)이고 $\tau \leq t$일 경우에는 Berlekamp-Massey 알고리즘이나 확장된 Euclidean 알고리즘을 이용하면 s로부터 오류벡터 $\boldsymbol{e}$를 도출할 수 있게 된다. 도출되는 오류벡터 $\boldsymbol{e}$ 내에서 0이 아닌 오류값(error magnitude)을 가지는 구성요소 $e_i \neq 0$의 위치 i를 오류위치(error location)라고 하며, 이것이 결국 오류가 어디에서 발생 했는지에 대한 근거가 된다. $s = \boldsymbol{e} \cdot \mathbf{H}^T$에서 오류벡터 내의 오류값이 0인 구성요소는 s를 생성하는 데에 아무 영향을 미치지 않기 때문에, 예를 들어 e_1 이외의 다른 구성요소가 모두 0이면 $s = \boldsymbol{e} \cdot \mathbf{H}^T = e_1 \cdot \boldsymbol{h}_1$이 된다. 이 사항을 공모자 추적 알고리즘에 대응시키면, 결국 오류위치가 공모자에 대한 신원을 파악하게 해 준다. 즉, 오류위치 $i_l \in U = \{i_1, i_2,$

..., i_τ }, (l = 1, 2, ...,. τ)에 대해서 오류값은 e_{il} (= $\beta_l b_{il}^{-1}$) ≠ 0. 여기서 i_l은 b_{il}을 만드는 데에 사용된 행렬 **H**의 위치 i_l과 일치하게 된다. 특히, 발생된 오류의 개수는 전체 공모자 수와 일치한다.

Ⅲ. 공모자 추적기법의 임계치와 조합론적 특성

t+1이상의 공모가 이루어질 경우는 결국 t+1이상의 오류가 발생한 경우인데, 오류수정부호의 복호화 오류(decoding error)에 의해서 어떤 공모자는 색출이 되지만 반면에 무고한 사용자(innocent user)가 공모자로 오인될 수가 있게 된다. 따라서, 임계치 t는 공모자 추적 알고리즘이 정확히 동작하기 위한 일종의 상계(upper bound)를 의미한다. 이번 장에서는 2장에서 소개된 간소화된 Boneh-Franklin 기법[5]을 기반으로, 그룹의 규모에 관계 없이 동일한 복호화 키를 만들어낼 수 있는 그룹의 존재 여부 그리고 그 개수 등을 파악해 봄으로써 선형부호를 이용한 공개키 공모자 추적기법의 조합론적인 특성을 분석해 본다.

3.1 표준배열과 공모자 그룹

정리 1에서는 불법 복호화 키 s가 주어졌을 때, 그것과 동일한 키를 만들어 낼 수 있는 사용자 그룹의 총 개수를 표준배열(standard array)을 기반으로 계산한다.

[정리 1] t명 이하의 공모자에 의해서 생성된 불법 복호화 키 s가 주어졌을 때, 그것과 동일한 키를 생성할 수 있는 모든 가능한 공모자 그룹의 총 개수는 q^k개이다.

<증명> 불법 복호화 키 s는 2장의 끝에서 언급한 것처럼 오증벡터이기 때문에 특정 오류벡터와 패러티 검사행렬 **H**의 곱에 의해서 생성이 된다. 부연하면, 해당 오류벡터는 q^{n-k}개의 coset으로 구성된 표준배열의 특정 coset과 연계된 coset leader가 된다. 표준배열의 특성상, 동일 coset에 속하는 모든 벡터는 해당 coset의 coset leader와 동일한 오증벡터를 가진다. 따라서, 1개의 coset은 q^k개의 벡터로 구성이 되기 때문에, 동일한 키를 생성할 수 있는 모든 가능한 공모자 그룹의 총 개수는 q^k개이다.

Coset leader인 오류벡터에서 0이 아닌 값을 가지는 오류의 위치는 공모자의 신분을 나타낸다. 즉, **H**의 열 벡터(column vector)들의 선형결합(linear combination)의 형태인 불법 복호화 키 s는 공모자 추적 알고리즘의 입력자료가 되며, 그것의 출력자료는 해당 열 번호(column number)가 된다. 또한, 동일한 coset에 속하는 오류벡터이외의 다른 벡터들, 즉 코드워드에 오류벡터가 첨가된 벡터들의 경우도 0이 아닌 값을 가지는 위치가 결국 가능성 있는 공모자들의 신분으로 간주되어질 수 있다. 따라서, 동일한 coset에 속하는 각각의 벡터들에 해당하는 공모자들의 신분들의 집합을 U_i (i = 1, 2, ... q^k)라고 정의하면 정리 1에서 언급한 동일한 키를 만들어 낼 수 있는 그룹들은 대부분의 경우 서로 공통된 공모자들이 포함될 수 있는 즉, $U_i \cap U_j \neq \varnothing$ ($i \neq j$)한 경우에 해당한다. 이와 관련하여 본 연구의 기본적인 취지는 $U_i \cap U_j = \varnothing$ ($i \neq j$)한 조건을 만족하는 경우가 존재하는 지를 확인하는 데에 있다.

동일한 복호화 키를 만들 수 있는 다음과 같은 조건을 만족하는 상호 독립적인 2개의 사용자 집합 U와 V가 존재한다고 가정하자.

$$U \text{ and } V \text{ such that } U \cap V = \varnothing, \quad |U| \leq t \text{ and } |V| \geq t+1 \tag{4}$$

만약, U와 V가 동일한 불법적인 복호화 키를 생성할 수가 있다면, 오류수정부호의 복호화 작업을 통해서는 항상 집합 U에 속한 사용자들을 공모자로 판별하게 된다. 이것이 의미하는 바는, 비록 V가 실제 공모자들의 집합이라 할지라도 복호화 작업은 U를 공모자들의 집합으로 오인할 수가 있고, 반대로 U가 실제 공모자들의 집합일 경우에 V 역시 공모자로 볼 수 있는 근거가 있기 때문에 그들, 즉 U가 공모자로 누명을 쓰고 있다고 주장을 해도 이를 반박할 근거가 없게 된다. 따라서, 임계치가 가지는 의미가 단지 추적 알고리즘의 복호화 오류가 발생하지 않게 하기 위한 것 이상의 의미를 가지고 있음을 확인한다. 다음 절에서 특정 조건 하에서 동일한 복호화 키를 가지는 2개의 상호 독립적인 사용자 집합이 존재함을 증명함으로써, 이와 같은 분쟁의 여지가 있음을 보일 것이다.

3.2 상호 독립적인 공모자 그룹

특정 그룹 U에 의해서 생성된 복호화 키가 주어

졌을 때, 그와 동일한 키 값을 생성할 수 있는 그룹 $V(U \cap V = \varnothing)$의 존재 가능성에 대해서 논의하기로 한다. 먼저, 특정 공모자 그룹이 생성할 수 있는 불법 복호화 키는 식 2에서와 같이 그들이 가지고 있는 각자의 개별 복호화 키를 $\beta_1 + \beta_2 + ... + \beta_\tau = 1$을 만족하게 조합하면 된다. 따라서, β_i의 값의 선정방식에 따라서 다양한 불법 복호화 키를 만들 수가 있다. $\tau \leq t$ 인 경우에 $U = \{ i_1, i_2, ..., i_\tau \}$를 공모자들의 집합이라 하고 다음을 정의한다.

$$\beta_l = \frac{b_{i_l}}{b_{i_1} + b_{i_2} + ... + b_{i_\tau}}, \ (l = 1, 2, \cdots, \tau) \tag{5}$$

위의 값을 $s = (s_1, s_2, ..., s_{n-k}) = \beta_1(b_{i1}^{-1}) \cdot \boldsymbol{h}_{i1} + \beta_2(b_{i2}^{-1}) \cdot \boldsymbol{h}_{i2} + ... + \beta_\tau(b_{i\tau}^{-1}) \cdot \boldsymbol{h}_{i\tau}$에 대입하면, 공모자들은 다음과 같은 불법 복호화 키 s를 생성할 수가 있다. 이때, $\beta_1 + \beta_2 + ... + \beta_\tau = 1$ 역시 만족된다.

$$s = b_U^{-1} \cdot \boldsymbol{h}_U, \ where \ b_U = b_{i1} + b_{i2} + \cdots + b_{i\tau}, \ (\boldsymbol{h}_U = \boldsymbol{h}_{i1} + \boldsymbol{h}_{i2} + \cdots + \boldsymbol{h}_{i\tau}) \tag{6}$$

즉, $\boldsymbol{e}_x = (e_1^{(x)}, e_2^{(x)}, \cdots, e_n^{(x)})$를 x번째 $(1 \leq x \leq q^{n-k})$ coset의 오류벡터라 하고, 오류위치 $i_l \in U$ (l = 1, 2, $\cdots$, τ)에 대해서 $e_{il}^{(x)} = \beta_l(b_{il}^{-1}) = b_U^{-1} \neq 0$을 만족하기 때문에 $s = \boldsymbol{e}_x \cdot \mathbf{H}^T$가 성립된다. 반대로, $V = \{ i_1', i_2', \cdots, i_\omega' \}$을 $U \cap V = \varnothing$그리고 $\omega \geq t+1$을 만족하는 임의의 무고한 사용자들의 집합이라고 하고, $b_V = b_{i1'} + b_{i2'} + \cdots + b_{i\omega'}$ 그리고 $\boldsymbol{h}_V = \boldsymbol{h}_{i1'} + \boldsymbol{h}_{i2'} + \cdots + \boldsymbol{h}_{i\omega'}$일 경우에 $s' = b_V^{-1} \cdot \boldsymbol{h}_V$라고 하자. 이때, $s = s'$이 성립하기 위해서는 다음이 만족되어야 한다. 즉, $s = \boldsymbol{e}_x \cdot \mathbf{H}^T$이기 때문에 $\boldsymbol{e}_x$와 동일한 coset에 속하는 어떤 $(\boldsymbol{c}_y + \boldsymbol{e}_x)$에 대해서 $s' = (\boldsymbol{c}_y + \boldsymbol{e}_x) \cdot \mathbf{H}^T$, ($1 \leq y \leq q^k$)로 표현된다. $\boldsymbol{e}_x$와 $\boldsymbol{c}_y + \boldsymbol{e}_x$에서 0이 아닌 구성요소의 위치 값이 각각 U와 V에 속하는 사용자들을 지칭하기 때문에, $U \cap V = \varnothing$의 의미는 $\boldsymbol{e}_x$에서 0이 아닌 구성요소의 위치 값에 해당하는 $\boldsymbol{c}_y + \boldsymbol{e}_x$에서의 구성요소는 0이어야 한다. 즉, $\boldsymbol{d}_{xy} = \boldsymbol{c}_y + \boldsymbol{e}_x = (d_1^{(xy)}, d_2^{(xy)}, \cdots, d_n^{(xy)})$라고 정의하면 $e_i^{(x)} \neq 0$을 만족하는 모든 $i \in U$에 대해서 $d_i^{(xy)} = 0$이 된다. 정리 2는 $t \leq k$조건 하에서 해당 오증벡터 s와 s'이 동일한, $U \cap V = \varnothing$을 만족하는 U와 V가 존재한다는 것을 보인다. 소수 q와 $t \leq k$에 대해서, GF(q)상의 t개의 오류를 수정할 수 있는 (n, k) Reed-Solomon 부호를 고려한다.

[정리 2] GF(q) 상의 t ($t \leq k$)개의 오류를 수정할 수 있는 (n, k) Reed-Solomon 부호에서, 오류벡터 $\boldsymbol{e}_x = (e_1^{(x)}, e_2^{(x)}, \cdots, e_n^{(x)})$를 포함하는 x번째 coset에는 $e_i^{(x)} \neq 0$을 만족하는 모든 $i \in U$에 대해서 i번째 구성요소가 모두 0인 벡터 $\boldsymbol{c}_y + \boldsymbol{e}_x$가 적어도 한개 이상 존재한다.

<증명> $\boldsymbol{c}_y + \boldsymbol{e}_x$의 $i(\in U)$번째 구성요소가 0이되기 위해서는 코드워드 $\boldsymbol{c}_y$의 i번째 구성요소와 $\boldsymbol{e}_x$의 i번째 구성요소의 합이 0이되어야 한다. $\mathbf{G}$를 기반이 되는 부호의 생성행렬이라고 하면, 코드워드 $\boldsymbol{c}_y$의 i번째 구성요소는 $\boldsymbol{m} \cdot \boldsymbol{g}_i$가 된다. 이때, $\boldsymbol{g}_i$는 $\mathbf{G}$의 i번째 열 벡터이다. 다음과 같은 k개의 미지수 그리고 τ개의 방정식으로 구성된 선형 시스템 $\boldsymbol{m} \cdot \boldsymbol{A} = (e_{i1}^{(x)}, e_{i2}^{(x)}, \cdots, e_{ir}^{(x)})$을 정의한다. 여기서 $\tau \leq t$ 그리고 $U = \{ i_1, i_2, \cdots, i_\tau \}$에 대해서 $\boldsymbol{A} = [\boldsymbol{g}_{i1}, \boldsymbol{g}_{i2}, \cdots, \boldsymbol{g}_{i\tau}]$. 행렬 $\mathbf{G}$의 임의의 k개의 열벡터는 Vandermonde 행렬이기 때문에 그 k개의 열 벡터는 선형 독립이다. 따라서, Rank(**A**) = $\tau \leq k$. 결국 선형 시스템의 해가 적어도 한 개 이상이 존재한다.

위의 정리 2를 만족하는 코드워드와 오류벡터를 컴퓨터 검색을 통해서 찾은 결과가 아래의 예 1에 나타나 있다. 이 예제는 정리 2의 정당성을 보여주는 예제이지, 실제 구현에 사용될 수 있는 것은 아니다.

[예 1] t=5개의 오류를 수정할 수 있는 GF(17) = { 0, 1, a^1, a^2, ..., a^{15} } 상의 (16, 6) Reed-Solomon 부호와 그에 대응되는 생성 다항식

$$\begin{aligned} g(x) &= (x - a^1)(x - a^2)(x - a^3)(x - a^4)(x - a^5)(x - a^6)(x - a^7) \\ &\quad (x - a^8)(x - a^9)(x - a^{10}) \\ &= x^{10} + a^6x^9 + a^{10}x^8 + a^9x^7 + a^{11}x^6 + a^5x^5 + a^6x^4 \\ &\quad + a^8x^3 + a^{11}x^2 + a^2x + a^7 \end{aligned}$$

을 고려해 보자. 이때, a=3는 원시원소가 된다. 생성행렬 $\mathbf{G}$는 생성 다항식 $g(x)$를 기반으로 구축되어질 수가 있고, 또한 이에 대응되는 패러티 검사행렬은 다음과 같다.

$$\mathbf{H} = \| a^{ji} \|, \ j = 1, 2, \cdots, 10 \ (i = 0, 1, 2, \cdots, 15)$$

.

본 예제의 부호는 조건 $t \leq k$을 만족한다. 시스템 비밀키는 $\boldsymbol{a} = (a_1, a_2, a_3, \cdots, a_9, a_{10}) = (a^3, a^{14}, a^4, a^{12}, a^5, a^{16}, a^9, a^{10}, a^2, a^3)$ 그리고 U = { 1, 2, 3,

4, 5 } 라고 하자. 이 경우는 $t = \tau = 5$을 의미한다. 이때, $b_1 = a^2$, $b_2 = a^3$, $b_3 = a^{11}$, $b_4 = a^{10}$, $b_5 = a^7$ 그리고 $b_U^{-1} = a^{15}$이기 때문에 e_x = (a^{15}, a^{15}, a^{15}, a^{15}, a^{15}, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0)가 된다. m = (a^{15}, 1, a^7, a^{12}, a^{15}, a^{14})일 경우에, e_x에 대응되는 코드워드는 $c_y = m \cdot G$ = (a^{15}, a^{15}, a^{15}, a^{15}, a^{15}, a^2, 0, a^4, a^9, a^3, a^{12}, 0, a^{11}, a^3, a^8, a^5)이다. e_x와 c_y+e_x는 동일한 coset에 속하기 때문에, 동일한 오증벡터를 가지게 된다. 따라서, V = { 6, 8, 9, 10, 11, 13, 14, 15, 16 } 의 복호화 키는 U의 복호화 키와 동일하게 된다.

2개의 상호 독립적인 사용자 집합 U와 V가 주어졌을 때, [정리 3]은 만약 둘 중의 하나가 t이하의 규모라면, 다른 하나는 항상 t+1이상의 규모임을 증명하고 있다.

[정리 3] 만약 $|U| \leq t$ 이면, 항상 $|V| \geq t+1$.

<증명> 만약 $|U| \leq t$ 조건하에서 $|V| \leq t$ 라고 가정하자. e와 $c+e$를 각각 U와 V에 해당하는 동일 coset에 속하는 2개의 벡터라고 할때, 이를 해밍무게로 표시하면 $wt(e) \leq t$와 $wt(c+e) \leq t$가 된다. 따라서, $wt(c) \leq wt(e) + wt(c+e) \leq 2t$ 이 성립하게 된다. 하지만, 코드워드의 최소거리가 $2t+1$ 이상이 되어야 하기 때문에, 가정이 모순이 된다.

Ⅳ. 결 론

선형부호를 이용한 공개키 기반 공모자 추적 기법에서의 임계치는 해당 추적 알고리즘이 정상적으로 동작하기 위한 공모 규모에 대한 상계이다. 만약, 공모 규모가 임계치 값을 넘어서게 되면 공모자 추적 알고리즘을 통해서는 공모에 참여한 모든 공모자들을 유일하게 색출해 낼 수가 없다. 본 논문에서는 임계치 값을 넘어서는 공모가 발생했을 경우에, 어떠한 조건 하에서 무고한 사용자들의 집합이 오류수정부호의 복호화 오류의 결과로 인하여 공모자로 오인될 수 있는지를 분석하였다. Reed-Solomon 부호를 이용한 공모자 추적 기법과 연관하여, 동일한 복호화 키를 가지는 2개의 상호 독립적인 사용자 집합을 구축하는 방식과 그 그룹의 규모를 논의하였다. 현실적으로, 우리가 임계치의 크기를 높게 잡는다 할지라도 어떤 그룹이 공모에 참여했는지에 대한 분쟁의 가능성은 항상 존재하게 된다.

참 고 문 헌

[1] R. E. Blahut, *Theory and Practice of Error Control Codes,* Addison-Wesley, 1984.

[2] D. Boneh, M. Franklin, An efficient public key traitor scheme, in: Proc. Crypto'99, *Lecture Notes in Computer Science 1666*, Springer Verlag, 1999, pp. 338~353.

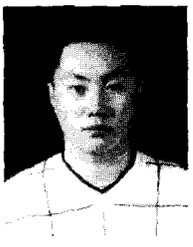
[3] B. Chor, A. Fiat, M. Naor, Tracing traitors, in: Proc. Crypto'94, *Lecture Notes in Computer Science 839,* Springer Verlag, 1994, pp.257~270.

[4] V. Guruswami, M. Sudan, Improved decoding of Reed-Solomon and algebraic-geometry codes, *IEEE Trans.* Inform. Theory, vol.45, no.6, 1999, pp.1757~1767.

[5] K. Kurosawa, T. Yoshida, Linear code implies public-key traitor tracing, in: Proc. *Public Key Cryptography*, 2001.3

〈著者紹介〉

임 정 미 (Jeong-Mi Lim) 준회원
2000년 2월 : 단국대학교 전자계산학과 졸업 학사
2002년 2월 : 단국대학교 전자계산학과 석사
2002년 3월~현재 : 단국대학교 전자계산학과 박사과정
<관심분야> 정보보호, 네트워크 보안



이 병 선 (Byoung-Sun Lee)
2002년 2월 : 단국대학교 전자계산학과 졸업 학사
2002년 3월~현재 : 단국대학교 전자계산학과 석사과정
<관심분야> 멀티미디어

박 창 섭 (Chang-Seop Park) 정회원
1983년 : 연세대학교 경제학과 졸업
1983년 : 한국 IBM 근무
1990년 : 미국 Lehigh Univ. 전자계산학 박사
1990년~현재 : 단국대학교 전자컴퓨터학부 교수
<관심분야> 부호이론, 암호학